Amendments to the Specification:

Please replace the paragraph beginning at **page 3, line 9**, with the following rewritten paragraph:

> Software programs written in the popular platform-independent Java (the platform-independent programming language from Sun Microsystems, Inc., Java is a trademark of Sun Microsystems, Inc.) language require that the Java classes be "packaged" for distribution but the package does not contain persistent information so once Java software is installed on a client computer, all information about it is lost. It is impossible to tell what the version number is, where it came from, or whom the author is. Additionally, the current network distribution methods make it difficult to digitally sign a Java package for security purposes.

Please replace the paragraph beginning at **page 4, line 7**, with the following rewritten paragraph:

> A software package manager uses a distribution unit containing components for a software package and a manifest file that describes the distribution unit to manage the installation, execution, and uninstallation of software packages on a computer. For installation, the package manager acquires the manifest file and parses it to learn if the software package depends on any additional components. The package manager resolves any dependencies by acquiring a distribution unit containing the needed component and installs the dependency's distribution unit as described below. Because dependencies can be nested within dependencies, the package manager recursively processes all the dependencies before finishing the installation of the software package that depends upon the additional components.

Please replace the paragraph beginning at **page 6, line 7**, with the following rewritten paragraph:

> The code store data structure maintained by the software package ~~manage~~ manager contains information about the installed software such as version and installation location, and is used to resolve version discrepancies among software programs that share components. The code store data structure is used by the package manager to locate necessary component when the software is executed so that a component stored in one directory can be readily shared by software programs with components in different directories. Finally, the code store data structure eases the uninstallation process by centralizing all the information about installed components.

Please replace the paragraph beginning at **page 7, line 4**, with the following rewritten paragraph:

FIGs. 3A, 3B, 3C and 3D are ~~flowchart~~ flowcharts of methods to be performed by a client according to an exemplary embodiment of the package manager of the invention; and

Please replace the paragraph beginning at **page 12, line 4**, with the following rewritten paragraph:

A system level overview of the operation of an exemplary embodiment of the invention is described by reference to FIGs. 2A, 2B and 2C. The exemplary embodiment is implemented in [[an]] a wide-area networking environment 52 having a server computer, such as remote computer 49 and a user or client computer, such as local computer 20, all of which are shown in FIG. 1 and described in the previous section.

Please replace the paragraph beginning at **page 17, line 11**, with the following rewritten paragraph:

When the distribution unit file for a software package is loaded onto a computer for installation, the software package manager running in the computer acquires the manifest file for processing (step 301). In an embodiment in which the manifest file is distributed in a distribution unit file, the package manager acquires the distribution unit file and extracts the manifest file from the distribution unit file. The package manager checks the name and version of the software package contained in the manifest file against the code store data structure to determine if the software package has already been installed (step 303). If so, the package manager exits ~~(step 321)~~ (step 321).

Please replace the paragraph beginning at **page 18, line 3**, with the following rewritten paragraph:

If there are dependencies, the package manager checks the code store data structure to determine if the dependencies are installed on the local computer (step 327). If not, the package manager uses information stored in the manifest file about the dependencies to locate a source for the dependencies. The package manager [[than]] then acquires a dependency from the source specified in the manifest file (step 329). In one embodiment, the source is a remote server designated by a uniform resource locator (URL) path name. In an alternate

embodiment, the source is a server on a local area network and the path name is a network drive.

Please replace the paragraph beginning at **page 20, line 5**, with the following rewritten paragraph:

If the newly stored component is older [[that]] than the previously stored component (step 349) and the software package does not indicate a version dependency (step 351), the package manager removes the older version from the newly created directory (step 353) and updates the code store data structure entry for the newly installed software package to point to the newer version (step 355). As before, if there is a version dependency noted, the package manager does nothing to the older version (step 347).

Please replace the paragraph beginning at **page 22, line 5**, with the following rewritten paragraph:

In one embodiment, the package manager scans every entry in the code store data structure to determine if another [[the]] entry for another software package references the local directory holding the component in question. In a first alternate embodiment, the package manager creates and maintains a tree structure of all shared components, so it can quickly determine if the component is shared with another software package. In a second alternate embodiment, the component is not deleted at the time the software package is uninstalled but instead a "garbage collection" routine is periodically run by the package manager. The garbage collection routine uses the code store data structure to determine if a component is referenced by any of the software packages installed on the local computer. Those components which are not referenced are then deleted.

Please replace the paragraph beginning at **page 22, line 17**, with the following rewritten paragraph:

Fig. 4 illustrates an exemplary embodiment of an entry in the code store data structure 400 suitable for use by the methods of the exemplary embodiments of the package manager described above. Each entry contains, at a minimum, five fields: a name field 401 for the distribution unit, a version field 403 for the distribution unit, a component field 405 that contains a list of the components in the distribution unit, a location field 407 that contains the location of the components on the client computer, and a source field 409 that contains a pointer to the source of the distribution unit, such as a URL for a downloaded distribution unit. If a component is a platform-specific ("native code") file, such as a Microsoft Windows DLL, the file name ~~is the component~~ is stored in the

component field. If a component is a package of Java classes, the component field contains the name of the Java package. In the case of a Java package, the code store entry has the following additional fields: a component version field 411 (which may be different from the version of the distribution unit; both are used by the package manager in resolving version dependencies), and a component type field 413 that indicates what type of Java classes the package contains, i.e., system, application, etc., both shown in phantom in ~~FIG. 4~~ FIG. 4. An optional data signature field 415, also shown in phantom, contains a digital signature affixed to the distribution unit, if it was signed. As [[well]] will be familiar to one of skill in the art, the digital signature itself can contain security attributes about the package. The security attributes can be used by the package manager to prevent versions updates from a source other than the signer. Furthermore, a vendor may include software packages from third parties as dependencies in the distribution unit and the package manager uses the digital signatures on each dependent package to direct the corresponding components into different local directories.

Please replace the paragraph beginning at **page 32, line 10**, with the following rewritten paragraph:

The terminology used in this application ~~with respect to~~ is meant to include all hardware and software platforms. Therefore, it is manifestly intended that this invention be limited only by the following claims and equivalents thereof.